SISTEMUL DE TIPURI AL LIMBAJULUI PASCAL

1. Tipuri predefinite

a) numerice: integer, real;
b) nenumerice: boolean, char.

2. Tipuri scalare definite de programator

a) tipuri enumerare:

type *sectii = (automatica, calculatoare, electronica, electrotehnica, energetica);*

b) subdomenii:

Tipuri noi pot fi definite ca **submulțimi** ale tipului **integer, char,** sau ale unui tip **enumerare.**

type *curenti_slabi = automatica .. electronica;*
cifre : '0' .. '9' ;
luna : 1 .. 12;

- Un tip subdomeniu este compatibil cu tipul de bază din care provine.
- Verificarea încadrării valorii pe care o primeşte o variabilă de tip subdomeniu se pote face, în general, doar la execuție.

3. Tipuri de date structurate

a) Tabloul;
b) Articolul;
c) Mulțimea;
d) Fişierul.

a) Tabloul.

- Implementează în Pascal proiecțiile finite.
- Forma generală a unei declarații de tip tablou:
  type *t =* ***array*** *[TI]* ***of*** *TE ;*
  TI – tipul indicelui (tip scalar, cu excepția tipului real);
  TE– tipul elementelor (tip oarecare).
- Tipul indicelui (adică inclusiv dimensiunea tabloului) sunt o parte a tipului tablou.
  type *t1 =* ***array*** *[1.. 10]* ***of*** *integer;*
  t2 = ***array*** *[1.. 20]* ***of*** *integer;*
  Tipurile t1 şi t2 sunt diferite, deci incompatibile.
- Nu se pot realiza proceduri generale care să accepte parametri tablouri de lungimi diferite.Soluția: **tablourile conforme** (în Pascal standard ISO). Se poate declara un parametru formal tablou fără a preciza limitele indicelui.

```
procedure   p ( var  a : array [ j .. s : integer]  of  real );
  var  i  : integer ;
  begin
    ----------
   for i:= j to s  do
     a[i]:= ------
   ----------------
  end;
```

Soluția nu este pe deplin satisfăcătoare (nu rezolvă problema caracterului static al variabilelor tablou).

În interiorul procedurii ***p*** nu se poate declara o variabilă tablou ***t*** a cărei dimesiuni să corepundă cu cea a argumentului:

```
var  t : array [ j..s ]  of  real ;
```

b) Articolul.

- Implementează produsele carteziene şi reuniunile variabile.

```
type persoana = record
                    Nume: array[1..30] of char;
                    Zi,luna,an:integer
                end;
var autor : persoana ;
```

- Accesul la câmpuri :

```
autor.nume := 'Popescu';
autor.zi := 5;   autor.luna := 4;    autor.an := 1951;
```

sau:

```
with autor do begin
      nume := 'Popescu';
      zi := 5;  luna := 4;   an := 1951
 end
```

- Reuniunile variabile se descriu ca articole cu variante.

```
type persoana = record
                    nume : array [1..30] of char;
                    zi,luna,an:integer;
                    case barbat : boolean of
                      true : (greutate,inaltime:real);
                      false : (maritat:booloean)
                end;
  var p : persoana;
```

Este una din facilitățile cele mai puțin sigure ale limbajului Pascal :

```
p.nume := 'Popescu';
p.zi := 30;      p.luna := 10;      p.an := 1950;
p.barbat := true;    p.greutate := 81;
p.barbat := false;
if p.maritat then -------
```

Accesul la câmpul *maritat* încă n-ar trebui permis.

c) Mulțimea (set).

- Sunt prevăzute operațiile de reuniune, intersecție, diferență, test de incluziune şi de apartenență.

- Tipul de bază trebuie să fie *scalar* şi nu poate fi *real*;

  type *t* = ***set of*** *integer;* incorect
  type *t* = ***set of*** *0..255;* corect

- Diferitele implementări limitează la o valoare maximă cardinalitatea tipului de bază.

d)Fişierul.

- Este o secvență de elemente, toate de acelaşi tip, tipul de bază al fisierului.

- Orice tip poate fi tipul de bază al unui fişier.

4. Pointeri

- Declararea unui tip pointer presupune precizarea tipului obiectelor referite.

- Constanta predefinită **nil** – valoarea pointerului care nu referă nici un obiect.

- Procedura standard **new** – creează obiecte anonime ce urmează a fi referite prin pointeri.

```
var p : ↑t;
-------
new(p);
```

- Procedura **dispose** – eliberarea unei zone anterior rezervată prin **new.**

```
dispose(p);
```

Aceasta poate să creeze, în Pascal, referințe fictive.

- Pointerii Pascal (spre deosebire de cei din C) nu pot referi decât obiecte anonime alocate dinamic (nu pot referi variabile declarate în program).

5. Echivalența tipurilor

- Diferite implementări au definit echivalența tipurilor Pascal în mod diferit.

- **Standardul Pascal ISO** – definiție a echivalenței de tip asemănătoare cu echivalența de nume, denumită *echivalență de declarație*.
  Tipuri compatibile – acelaşi nume sau descrise prin **aceeaşi specificare de tip.**

```
type t = record
                n: array [1..30] of char;
                i : integer;
         end;
     tx = record
                n : array [1..30] of char;
                i : integer;
         end;
    t1 = tx;
var
   x: t ;    y : tx;    z : t1;
```

Variabilele **x** şi **z** sunt compatibile(tipul lor este descris prin aceeaşi declarație **record**)

- În Pascal, tipurile subdomeniu sunt compatibile cu tipurile de bază din care provin.

SISTEMUL DE TIPURI AL LIMBAJULUI ADA

1. Tipuri predefinite

a) numerice: integer, short_integer,long_integer;
float, short_float, long_float.

b) nenumerice: character; boolean.

2. Tipuri scalare definite de programator

a) numerice.

- Se pot defini tipuri întregi sau reale, precizând limite ale câmpului de valori sau precizia minimă la reprezentare (în cazul numerelor reale).
  type *sub_suta* ***is range*** *0..99;*
  type *real* ***is*** *digits 7;*
  type *real_mic* ***is digits*** *7* ***range*** *0.0..100.0;*
- Se permite lucrul cu numere în virgulă fixă:
  type *sutimi* ***is delta*** *0.01* ***range*** *0.0..1.0;*

b) tipul enumerare.

Este similar cu cel cunoscut din Pascal:
type *sectii* ***is*** *(automatica, calculatoare, electrotehnica, electronica, energetica);*

3. Tipuri de date structurate

a) Tabloul;
b) Articolul.

a) Tabloul.

- Pot fi declarate static (mulțimea indicilor – cunoscută şi fixată la compilare).

type *nr_studenti* ***is array****(sectii)* ***of*** *integer;*
type *tab* ***is array****(1..10)* ***of*** *integer;*

- La definire, este suficient să se precizeze doar tipul de bază al indicilor.

type *nr_st* ***is array*** *(sectii* ***range****<>)* ***of*** *integer;*
type *matrice* ***is array*** *(integer* ***range****<>, integer* ***range****<>)* ***of*** *real;*
type *biti* ***is array*** *(integer* ***range****<>)* ***of*** *boolean;*

- La declararea unui obiect tablou corespunzător unui tip fără restricții trebuie specificate restricțiile pentru indici (chiar şi prin expresii a căror valoare se calculează în execuție).

nrst_curenti_slabi:nr_st(automatica..electronica);
tab:matrice(1..5,1..5);
masca:biti(1..n);
mat:matrice(1..n,1..m);

- Se permite realizarea de proceduri care să lucreze cu argumente de lungimi oarecare.

type *vect* ***is array****(integer* ***range****<>)* ***of*** *real;*
procedure *p(a:****in out*** *vect)* ***is***
temp:vect(a'first..a'last);
i:integer;
begin
---
for *i* ***in*** *a'first..a'last* ***loop***
temp(i):=a(i);
end loop*;*
---
end *vect;*

Atributele standard **first** si **last** returnează limita inferioară, respectiv superioară a indicelui.

b) Articolul.

- Implementează (ca şi în Pascal) produsele carteziene şi reuniunile variabile (la care sunt evitate neajunsurile în ceea ce priveşte siguranța).

```
type persoana is
        record
            nume:string(1..30);
            zi,luna,an:integer;
        end record;
```

- Prezența câmpului selector este obligatorie. În timpul execuției se verifică validitatea referirii unui câmp.

```
type persoana (barbat:boolean:=true) is
        record
            nume:string(1..30);
            zi,luna,an:integer;
           case barbat is
                when true =>greutate,inaltime:float;
                when false =>maritat:boolean;
          end case;
        end record;
```

- Valoarea câmpului selector poate fi modificată doar odată cu atribuirea unei noi valori întregului articol.

```
        pm:persoana;
        pf:persoana(false);
```

Instrucțiuni ilegale:

```
        pm.barbat:=false;
        pf.barbat:=true;
```

Instrucțiuni legale:

```
        pm:=(false,"ionescu",25,05,1958,true);
```

sau

```
        pm:=pf;
```

- Nu este nevoie să se precizeze o valoare inițială pentru câmpul selector.

4. Pointeri

- La declararea unui tip pointer se precizează tipul obiectelor referite (ca şi în Pascal). Există valoarea predefinită **null** şi alocatorul **new**.

  type *ref* ***is access*** *t;*
  referinta:ref;
  ---
  *referinta:=**new** t;*

- S-au luat mai multe măsuri pentru a evita referințele fictive:
  - Atunci când nu există nici un pointer care referă obiectele alocate, memoria este în mod automat eliberată.
  - Momentul eliberării este acela când se părăseşte domeniul tipului pointer corespunzător.

- Programatorul poate elibera,în mod excepțional, zone de memorie alocate dinamic cu ajutorul procedurii **unchecked_deallocation**(relocare nestânjenită).
  În acest fel pot să apară referințe fictive.

5. Echivalența tipurilor. Subtipuri şi tipuri derivate

- Echivalența tipurilor se defineşte strict conform declarației de nume. Orice declarație de tip introduce un tip nou.

```
type pret is range 0..integer'last;
type sub_suta is range 0..99;
        x:pret;
        y:sub_suta;
        ----------
        y:=x;       -- este ilegal
```

- Obiectele declarate ca *subtip* sunt compatibile cu tipul de bază.

```
subtype sub_suta is pret range 0..99;
                y:sub_suta;
        -------------------------
        y:=x     -- este legal, dar se verifică 0 ≤ y≤ 99.
```

- Tipurile derivate permit introducerea unui tip care să aibă toate proprietățile unui tip anterior declarat, dar care să fie considerat tip nou.

```
type nr_studenti is new pret;
        a:pret;
        b:nr_studenti;
-----------------------------
        a:=b;       --  este ilegal
```

Tipul **nr_studenti** moşteneşte toate însuşirile tipului **pret** (aceeaşi mulțime de valori, aceleaşi operații şi atribute). Ele reprezintă însă tipuri diferite.

COMPARAȚIE.
LIMBAJE PUTERNIC TIPIZATE

- **Ada este un limbaj puternic tipizat**:
  - toate verificările privind compatibilitățile de tip se realizează static, la compilare.

- **Pascal nu este puternic tipizat** :
  - Prin articolele cu variante pot interveni violări ale sistemului de tipuri;
  - La tablouri, verificarea încadrării unui indice în limitele impuse reprezintă o verificare de tip şi se realizează în execuție.
  - În cazul subdomeniilor se face, în execuție, verificarea compatibilității de tip.

```
type t = 0..100;
var a,b,c:t;
--------------
        a := b + c;
--------------
```

Secvența analogă Ada este:

```
type t is range 0..100;
a,b,c:t;
-------
        a := b + c;
--------
```

Definiția tipului *t* este echivalentă cu:

```
type anonim is new integer;
subtype t is anonim range 0.100;
```

- **C nu este un limbaj puternic tipizat:**

- Obiecte de tip *char* pot să apară în expresii aritmetice (*char* este un întreg mai mic);

- Nediferențiere între valori întregi şi logice;

- Neefectuarea unor verificări de tip, de exemplu încadrarea valorii unei variabile de tip enumerare în mulțimea constantelor tipului;

- Definiția **union** este chiar mai nesigură decât în Pascal;

- *Pointerul la void* permite violarea compatibilităților de tip;